

## <JSP程序设计> (第2版) 第2章

# JSP页面与JSP标记

Power point 制作：张跃平 耿祥义

配合< **JSP**程序设计(第2版)代码>一起使用

# JSP 第2章 导读

## 主要内容

- **JSP**页面的基本结构
- 变量和方法的声明
- **Java**程序片
- 表达式
- **JSP**中的注释
- **JSP**指令标记
- **JSP**动作标记

## 难点

- **Java**程序片的运行原理
- **include**指令标记与**include**动作标记

## 关键实践

- 编写一个**JSP**页面，让该**JSP**页面包含**5**种基本的元素
- 编写含有**JSP**指令标记的**JSP**页面
- 编写含有**JSP**动作标记的**JSP**页面



## § 2.1 JSP页面的基本结构

**JSP**页面可由**5**种元素组合而成：

- ① 普通的**HTML**标记符；
- ② **JSP**标记，如指令标记、动作标记；
- ③ 变量和方法的声明；
- ④ **Java**程序片；
- ⑤ **Java**表达式；

**JSP**页面的运行原理：

- \* 把JSP页面中普通的HTML标记符号，交给客户的浏览器执行显示。
- \* JSP标记、数据和方法声明、Java程序片由Tomcat服务器负责执行，将需要显示的结果发送给客户的浏览器。
- \* **Java**表达式由**Tomcat**服务器负责计算，将结果转化为字符串，交给客户的浏览器负责显示。

# 例子1

例子1中，example2\_1.jsp页面包含了5种元素，页面效果如图2.1。

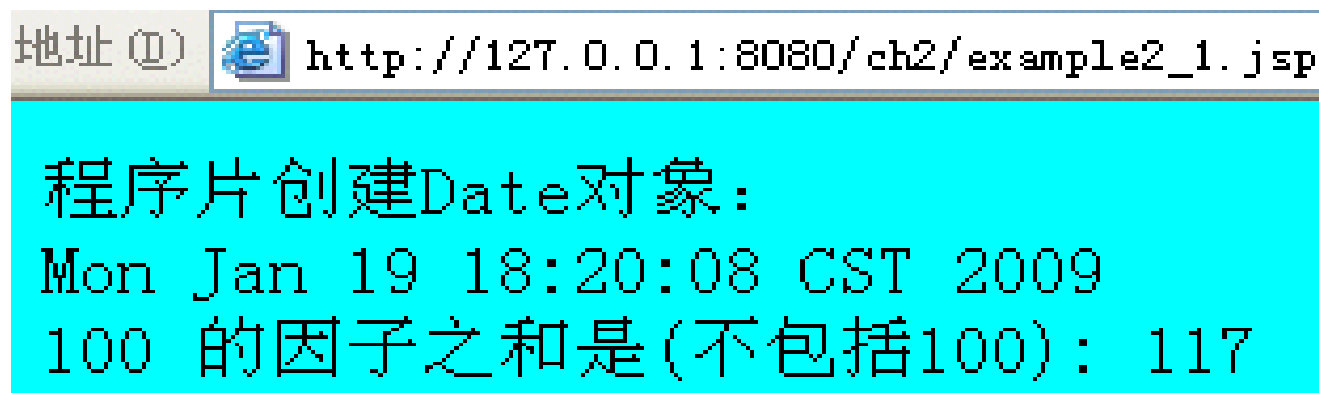


图 2.1 包含了 5 种元素的 JSP 页面

## 1. 在**JSP**页面的标记符

“**<%!**”和 “**%>**”

之间声明的变量称作JSP页面的成员变量。

例2-2

## 2. 在**JSP**页面的标记符

“**<%!**”和 “**%>**”

之间声明方法。该方法在整个JSP页面有效。

例2-3

## 例子2

例子2利用成员变量被所有用户共享这一性质，实现了一个简单的计数器，页面效果如图2.2。

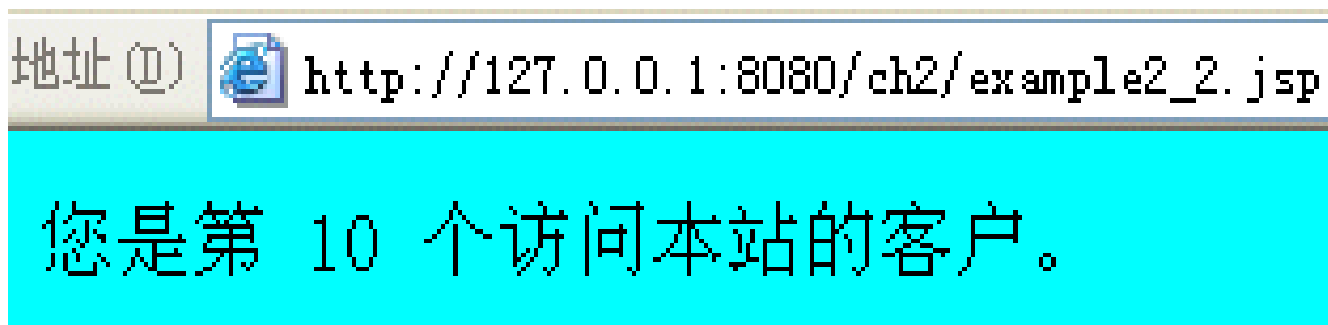



图 2.2 JSP 页面的成员变量

## 例子3

例子3中，example2\_3.jsp在“<%!”和“%>”之间声明定义了两个方法：getArea(double a)和getLength(double a)，在程序片中调用这两个方法，分别计算圆的面积和周长。example2\_3.jsp页面效果如图2.3。

地址 (D)  http://127.0.0.1:8080/ch2/example2\_3.jsp

调用getArea方法计算半径是100.0的圆的面积: 31415.926535897932  
调用getLength方法计算半径是50.0的圆的周长: 314.1592653589793

图 2.3 JSP 页面中方法的声明与调用

## § 2.3 Java程序片

- 在 “<%”和 “%>”之间插入**Java**程序片。
- 程序片中声明的变量称为**JSP**页面的**局部变量**。
- 多个客户请求一个**JSP**页面时，**Java**程序片将**被执行多次**，分别在不同的线程中执行。例2-4,例2-5

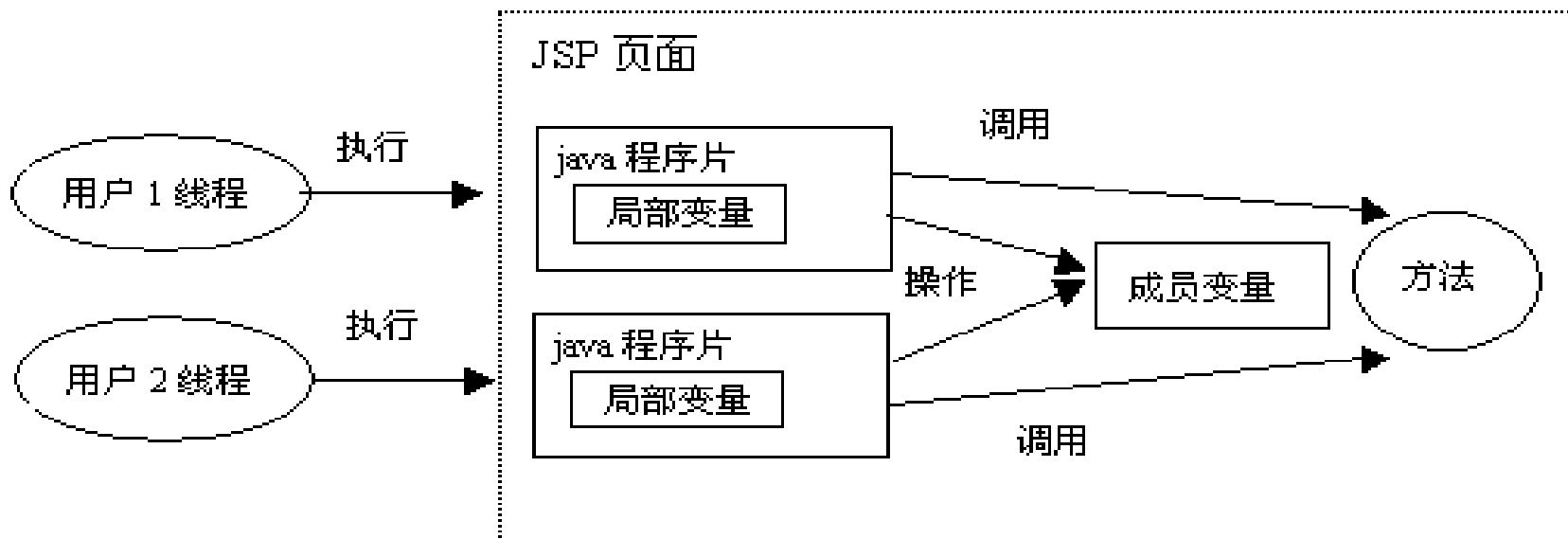


图 2.4 程序片的执行



## 例子4

例子4中，通过synchronized方法操作一个成员变量来实现一个简单的计数器。

```
example2_4.jsp
<%@ page contentType="text/html;Charset=GB2312" %>
<HTML><BODY>
    <%! int count=0; //被用户共享的count
        synchronized void setCount() { //synchronized修饰的方法
            count++;
        }
    %>
    <% setCount(); //程序片中调用同步方法
        out.println("您是第"+count+"个访问本站的用户");
    %>
</BODY></HTML>
```

## 例子5

例子5通过将程序片分割成几部分，来验证用户输入的E-mail地址中是否含有非法的字符，页面效果如图2.5。



图 2.5 分割程序片



## § 2.4 表达式

- “<%=”和 “%>”之间可以是一个Java表达式 。
- 表达式的值由服务器负责计算，并将计算结果用字符串形式发送到客户端显示。

### 例2-6

## 例子6

例子6计算表达式的值，页面效果如图2.6。

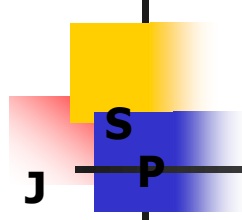
example2\_6.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<HTML><BODY bgcolor=cyan><FONT size=3>
<% double x=0.9,y=3;
%>
<P> Sin(<%=x%>)除以<%=y%>等于<%=Math.sin(x)/y%>
<p><%=y%>的平方是: <%=Math.pow(y,2)%>
<% x=19;
    y=32;
%>
<P><%=x%>乘以<%=y%>等于 <%=x*y%>
<P> <%=y%>的平方根等于 <%=Math.sqrt(y)%>
<P><%=y%>大于<%=x%>吗? 回答: <%=y>
</FONT></BODY></HTML>
```

地址 http://127.0.0.1:8080/ch2/example2\_6.jsp

Sin(0.9)除以3.0等于0.2611089698758278  
3.0的平方是: 9.0  
19.0乘以32.0等于 608.0  
32.0的平方根等于 5.656854249492381  
32.0大于19.0吗? 回答: true

图 2.6 计算表达式的值



## § 2.5 JSP中的注释

### 1. HTML注释格式:

**<!-- 注释内容 -->**

### 2. JSP注释格式:

**<%-- 注释内容 --%>**

注：程序片中的注释 //

**例2-7** 例子7中的JSP页面使用了HTML注释和JSP注释。

**example2\_7.jsp**

## § 2.6 JSP 指令标记

### 2.6.1 page 指令

page 指令用来定义整个JSP页面的一些属性和这些属性的值。

page 指令标记可以指定如下属性的值 **contentType**、**import**、**language**、**session**、**buffer**、**autoFlush**、**isThreadSafe**、**pageEncoding**。

属性值用单引号或双引号括起来。可以用一个page指令指定多个属性的值，也可以使用多个page指令分别为每个属性指定值。

page指令的作用对整个JSP页面有效，与其书写的位置无关，习惯把page指令写在JSP页面的最前面。

例如：

```
<%@ page 属性1="属性1的值" 属性2="属性2的值" .....%>
```

或

```
<%@ page 属性1="属性1的值" %>
```

```
<%@ page 属性2="属性2的值" %>
```

```
... ..
```

```
<%@ page 属性n="属性n的值" %>
```

定义JSP页面使用的脚本语言，该属性的值目前只能取“java”。

例如：

```
<%@ page language="java" %>
```

注：JSP页面默认有如上page指令。

该属性的作用是为JSP页面引入Java运行环境提供的包中的类，这样就可以在JSP页面的程序片部分、变量及函数声明部分、表达式部分使用包中的类。

例如：

```
<%@ page import="java.io.*", "java.util.Date" %>
```

注：JSP页面默认import属性已经有“**java.lang.\***”、“**javax.servlet.\***”等值。



## 2.6.1 page 指令\_contentType属性

**contentType** 属性值确定 **JSP** 页面响应的 **MIME** (**Multipurpose Internet Mail Extension**) 类型和**JSP**页面字符的编码。

例如：

```
<%@ page contentType="text/html;charset=GB2312" %>
```

```
<%@ page contentType="application/msword" %>
```

注：不允许两次使用**page** 指令给**contentType**属性指定不同的属性值。

例**2-8**

## 例子8

例子8中有两个JSP页面，其中的first.jsp页面使用page指令设置contentType属性的值是"text/html;charset=GB2312"，当用户请求first.jsp页面时，用户的浏览器启用HTML解析器来解析执行收到的信息；second.jsp页面使用page指令设置contentType属性的值是"application/msword"，当用户请求second.jsp页面时，用户的浏览器将启动本地的MS-Word应用程序来解析执行收到的信息，页面效果如图2.7（a）、2.7（b）。

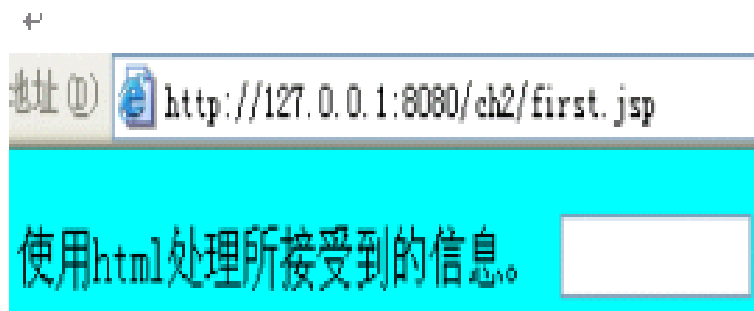


图 2.7（a）·HTML 解析器解析信息

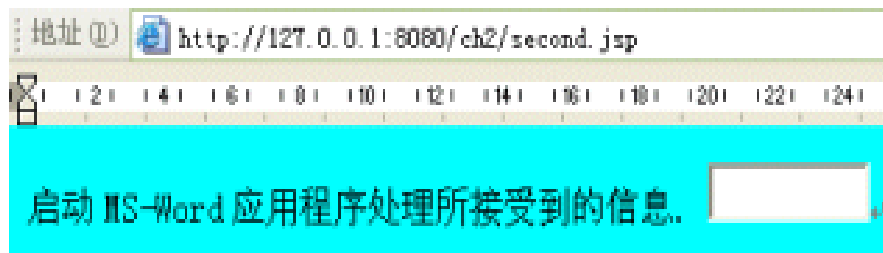


图 2.7（b）·MS-Word 应用程序解析信息

用于设置是否需要使用内置的session对象。session的属性值可以是true或false。session属性默认的属性值是true。

内置输出流对象out负责将服务器的某些信息或运行结果发送到客户端显示，buffer属性用来指定out设置的缓冲区的大小或不使用缓冲区。例如：

```
<%@ page buffer= "24kb" %>
```

buffer属性的默认值是8kb 。

buffer属性可以取值“none”，设置out不使用缓冲区。

参考 § 4.4 详细讲解out对象



## 2.6.1 page 指令\_ autoFlush属性

autoFlush属性:

指定out的缓冲区被填满时，缓冲区是否自动刷新。

注： autoFlush属性的默认值是true。

## 2.6.1 page 指令\_isThreadSafe属性

isThreadSafe属性:

用来设置JSP页面是否可多线程访问。

注: isThreadSafe属性的默认值是true。

**computer.jsp**

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page isThreadSafe="false" %>
<HTML><BODY>
  <%! int i=1; //被所有用户共享
  %>
  <%
    for(int k=1;k<=100;k++){
      out.println(i);
      i++;
    }
  %>
</BODY></HTML>
```

info属性的属性值是一个字符串，其目的是为JSP页面准备一个常用且可能需要经常修改的字符串。

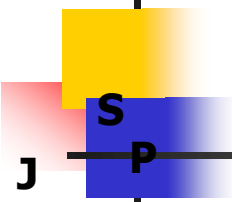
例如：

```
<%@ page info= "we are students" %>
```

注：可以在JSP页面中使用方法：

```
getServletInfo();
```

获取info属性的属性值。



## 2.6.2 include 指令标记

include指令标记的作用是在JSP页面出现该指令的位置处，静态插入一个文件。其语法格式如下：

**<%@ include file= "文件的URL " %>**

**注：**如果该文件和当前**JSP**页面在**同一Web**服务目录中，那么“文件的**URL**”就是文件的**名字**；

如果该文件在**JSP**页面所在的**Web**服务目录的一个子目录中，比如**fileDir**子目录中，那么“文件的**URL**”就是“**fileDir/**文件的**名字**”。

**注：**静态插入，就是当前JSP页面和插入的文件合并成一个新的JSP页面，然后JSP引擎再将这个新的JSP页面转译成Java文件。

**例2-9， 2-10**



## 例子9

例子9中的example2\_9.jsp页面使用include指令标记静态插入一个文本文件Hello.txt,此文件和当前example2\_9.jsp页面在同一Web服务目录中,页面效果如图2.8。

Hello.txt

```
<%@ page contentType="text/html;charset=GB2312" %>
```

很高兴认识你们!

nice to meet you.

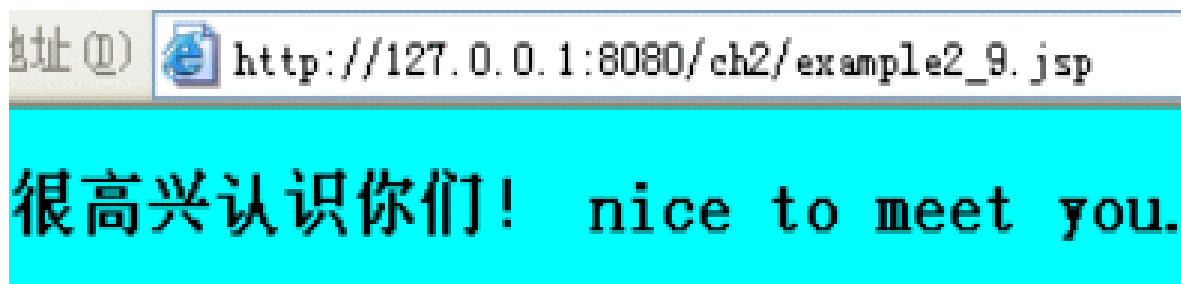


图 2.8 使用 include 指令标记

## 例子10

例子10中的JSP页面example2\_10.jsp使用include指令标记静态插入一个JSP文件computer.jsp。computer.jsp和example2\_10.jsp均保存在Web服务目录ch2中，example2\_10.jsp页面的效果如图2.10。

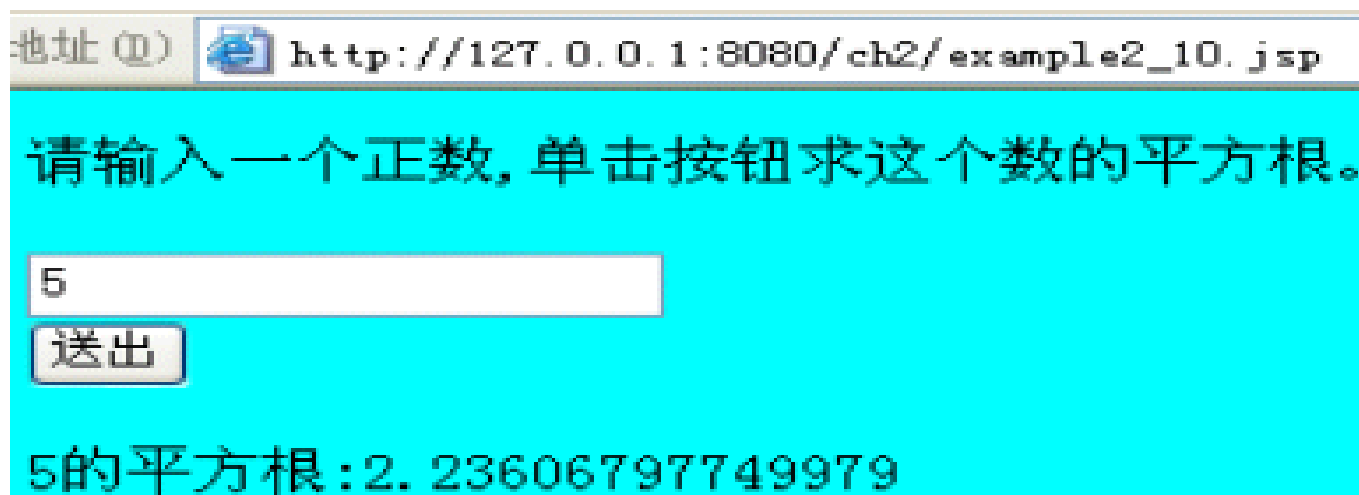


图 2.9- 使用 include 指令标记



## § 2.7 JSP 动作标记

### 2.7.1 include 动作标记

语法格式:

**<jsp:include page= "文件的URL"/>**

或

**<jsp:include page= "文件的URL">**

**param**子标记

**</jsp:include>**

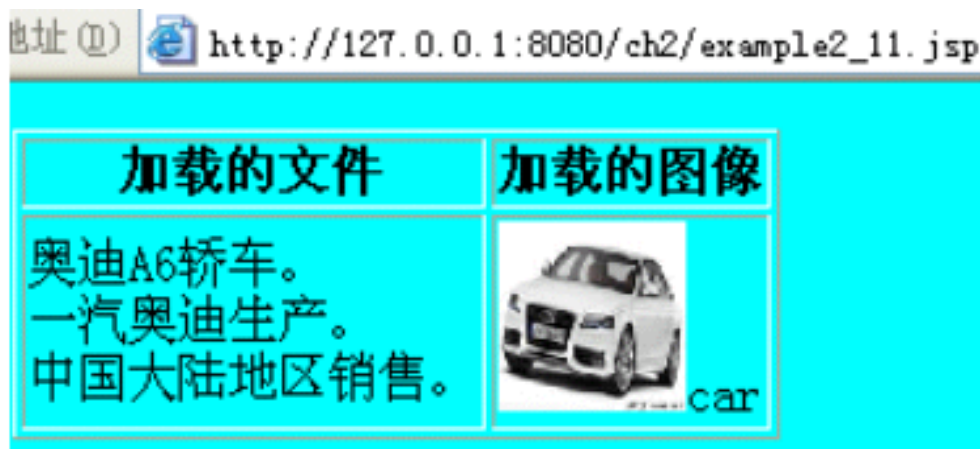
include动作标记告诉JSP页面动态加载一个文件。

**注:** include 动作标记是在JSP页面运行时才处理文件，被处理的文件在逻辑和语法上独立于当前JSP页面。

**例2-11**

## 例子11

例子11中的 **example2\_11.jsp** 页面动态加载两个文件：**imageCar.html**和**car.txt**。我们把**example2\_11.jsp**页面保存Web服务目录ch2中。**example2\_11.jsp**页面要动态加载的**imageCar.html**文件，以及**imageCar.html**文件所使用的图像文件**car.jpg**均保存在ch2中； **example2\_11.jsp**页面要动态加载的**car.txt**文件保存在ch2的子目录Myfile中。

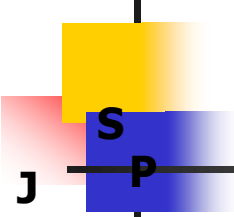


在浏览器的地址栏输入

**[http://127.0.0.1:8080/ch2/example2\\_11.jsp](http://127.0.0.1:8080/ch2/example2_11.jsp)**

访问example2\_11.jsp页面的效果如图2.10。

图 2.10 使用 include 动作标记



## 2.7.2 param动作标记

param标记以“名字-值”的形式为其它标记提供附加信息。

### 语法格式

**<jsp:param name=“名字” value=“指定给param的值” >**

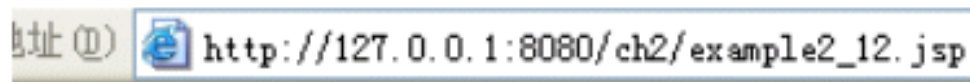
**注：** param 标记不能独立使用，需作为 jsp:include、jsp:forward、jsp:plugin 标记的子标记来使用。

**注：** 当该标记与 jsp:include 动作标记一起使用时，可以将 param 标记中的值传递到 include 动作标记要加载的文件中去，被加载的 JSP 文件可以使用 Tomcat 服务器提供的 request 内置对象获取 include 动作标记的 param 子标记中 name 属性所提供的值。

### 例2-12

## 例子12

例子12中, **example2\_12.jsp** 页面使用 **include** 动作标记动态加载文件 **tom.jsp**, 当 **tom.jsp** 文件被加载时获取 **example2\_12.jsp** 页面中 **include** 动作标记的 **param** 子标记中 **name** 属性的值 (tom.jsp 文件使用 Tomcat 服务器提供的 request 内置对象获取 param 子标记中 **name** 属性的值, 有关内置对象见后面的第4章)。



加载文件效果:

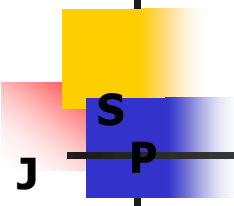
从1到300的连续和是: 45150

图 2.11· 动态加载文件计算连续和

在浏览器的地址栏输入

**http://127.0.0.1:8080/ch2/example2\_12.jsp** 访

问 example2\_12.jsp 页面的效果如图 2.11。



## 2.7.3 forward动作标记

该指令的作用是：从该指令处停止当前页面的继续执行，而转向执行page属性指定的JSP页面。

语法格式：

```
<jsp:forward page="要转向的页面" />
```

或

```
<jsp:forward page="要转向的页面" >
```

```
    param子标记
```

```
</jsp:forward>
```

**例2-13**

## 例子13

例子13中的**example2\_13.jsp**页面使用forward动作标记转向come.jsp页面，并向come.jsp页面传递一个数值。**example2\_13.jsp**和**come.jsp**页面保存在Web服务目录ch2中。在浏览器的地址栏输入：[http://127.0.0.1:8080/ch2/example2\\_13.jsp](http://127.0.0.1:8080/ch2/example2_13.jsp)访问**example2\_13.jsp**页面的效果如图2.12。

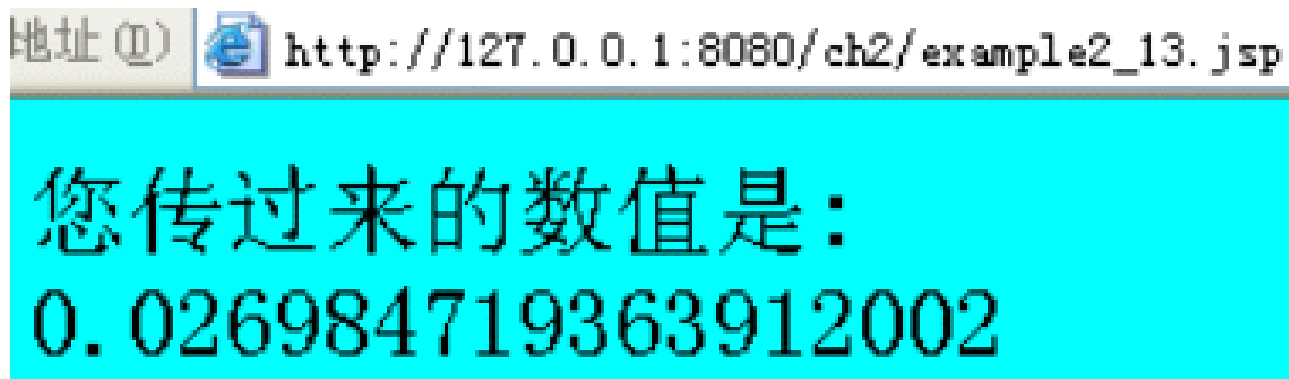
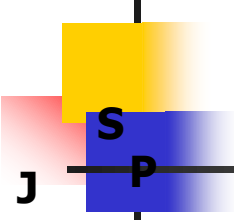


图 2.12 向转向的页面传递数据





## 2.7.4 plugin动作标记

该动作标记指示**JSP**页面加载**Java plugin**插件。该插件由用户负责下载，并使用该插件来运行**Java applet**小程序。语法格式：

```
<jsp:plugin type="applet" code="小程序的字节码文件"
```

```
  jreversion="java虚拟机版本号" width="小程序宽度值" height="小程序高度值" >
```

```
  <jsp:fallback>
```

提示信息：用来提示用户的浏览器是否支持插件下载

```
  </jsp:fallback>
```

```
</jsp:plugin>
```

### 例2-14

## 例子14

例子14, 假设有一个**Java applet**小程序, 主类字节码文件是**B.class**, 该文件存放在Web服务目录ch2中, 含有plugin动作标记的JSP文件**example2\_14.jsp**也存放在ch2中。

### example2\_14.jsp

```
<%@ page contentType="text/html; charset=GB2312" %>
```

```
<HTML><BODY>
```

```
<jsp:plugin type="applet" code="B.class" jreversion="1.2" width="200" height="260" >
```

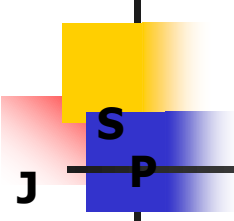
```
<jsp:fallback>
```

```
    Plugin tag OBJECT or EMBED not supported by browser.
```

```
</jsp:fallback>
```

```
</jsp:plugin>
```

```
</BODY></HTML>
```



## 2.7.5 useBean动作标记

该标记用来创建并使用一个Javabeen，是非常重要的一个动作标记，我们将在第4章详细讨论。Sun公司的倡导是：用HTML完成JSP页面的静态部分，用Javabeen完成动态部分，实现真正意义上的静态和动态分离。

参考第7章 详细讲解